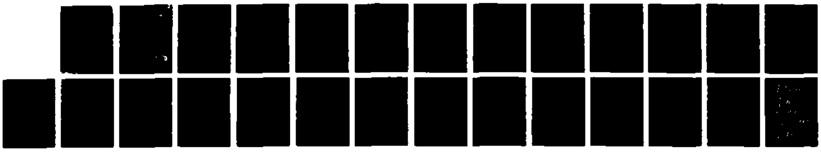
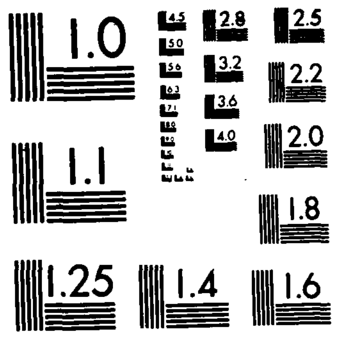
AD-A193 443 MACRO-FORT LEAVENWORTH IMPROVED KELLNER GRAPHICS INTERFACE PACKAGE (MACRO-FLIK)(U) ARMY TRADOC ANALYSIS COMMAND FORT LEAVENWORTH KS R H KAEDING MAY 88 1/1

UNCLASSIFIED TRAC-F-TM-1473 F/G 12/5 NL




1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

Technical Memorandum TRAC-F-TM-1473
May 1988

TRADOC Analysis Command-Fort Leavenworth (TRAC-FLVN)
Operations Directorate, Technology Applications Branch
Fort Leavenworth, Kansas 66027-5200

MACRO-FT LEAVENWORTH IMPROVED KELLNER GRAPHICS INTERFACE PACKAGE (MACRO-FLIK)

by

R.H. "Pete" Kaeding

ACN 48722

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; distribution is unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) TRAC-F-TM-1473		5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION TRAC-FLVN	6b. OFFICE SYMBOL (If applicable) ATRC-FOC	7a. NAME OF MONITORING ORGANIZATION		
6c. ADDRESS (City, State, and ZIP Code) Commander TRAC-FLVN (ATTN: ATRC-FOC) Ft Leavenworth, KS 66027-5200		7b. ADDRESS (City, State, and ZIP Code)		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c. ADDRESS (City, State, and ZIP Code)		10. SOURCE OF FUNDING NUMBERS		
		PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Macro-Ft Leavenworth Improved Kellner Graphics Interface Package (Macro-FLIK)

12. PERSONAL AUTHOR(S)
R. H. "Pete" Kaeding

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final Report	FROM 1-88 TO 5-88	1988 May	24

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This document is a user's manual for the Macro-Fort Leavenworth Improved Kellner Graphics Interface Package (Macro-FLIK). Macro-FLIK consists of a minimal set (as of this writing 16) of higher level graphics routines intended to provide the VAX application software developer with virtually no graphics background with an easy to use graphics package which will enable him/her to incorporate Ramtek graphics into any software application package. The software provides a means of incorporating those programming functions which lend themselves nicely to a graphics mode of processing; menuing, data display, and data modification.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT		21. ABSTRACT SECURITY CLASSIFICATION
[✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS		UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL R.H. "PETE" Kaeding	22b. TELEPHONE (Include Area Code) (913) 684-4261	22c. OFFICE SYMBOL ATRC-FOC

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

Table of Contents

Abstract

This document is a user's manual for the Macro-Fort Leavenworth Improved Kellner Graphics Interface Package (Macro-FLIK). Macro-FLIK consists of a minimal set (as of this writing 16) of higher level graphics routines intended to provide the VAX application software developer with virtually no graphics background with an easy to use graphics package which will enable him/her to incorporate Ramtek graphics into any software application package. The software provides a means of incorporating those programming functions which lend themselves nicely to a graphics mode of processing: menuing, data display, and data modification.

1. <u>Background.</u> The majority of the graphics display hardware used by the TRADOC Analysis Command (TRAC) is manufactured by Ramtek. A very low-level software package is provided by Ramtek to accompany this hardware. This package represents the "nuts and bolts" software and, as such, is not very user friendly. To simplify communications with the hardware, a slightly higher level set of routines, designed more with the user in mind, is preferred. Mr. Al Kellner, TRAC-White Sands Missile Range (TRAC-WSMR) attempted to fill this bill by developing what has come to be known within TRAC as the Kellner Graphics Interface Package (KGIP). This package, currently consisting of approximately 120 routines, is considerably more user friendly, and it provides the graphics programmer with a means of utilizing most of the Ramtek's capabilities. The Technology Applications BranchGraphics Team, Operations Directorate, TRAC-Ft Leavenworth (TAB-GT), confronted with several application software packages using KGIP variants, set out to improve the package in several ways.

a. At least seven variants of the package were known to exist. Our initial objective was to provide a single version package which would be compatible with all known application packages and all TRAC hardware configurations. To accomplish this, TAB-GT designed and developed a number of routines to make KGIP "intelligent" enough to dynamically determine the hardware configuration on which it's operating as well as that on which it was intended to operate. A byproduct of the evolution of this new package, Fort LeavenworthImproved KGIP (FLIK), was a much better organization of the package. This improved organization was primarily due to the consolidation of several families of routines (routines performing very nearly the same function) into what we call "super" routines. Other improvements incorporated into FLIK addressed KGIP deficiencies, omissions, or, in some cases, just represent additional sophistication. The end product, FLIK, provides the graphics programmer with a flexible, powerful, and relatively user friendly package.

b. Unfortunately, "user friendly" is "in the eye of the beholder." The typical applications programmer, faced with deadlines and capability requirements to satisfy, doesn't have the time to devote to graphics self-education. This led TAB-GT to the development of a package of higher level graphics routines. This package is intended to provide the application software developer with an easy to use, minimal set of routines which will enable him to incorporate VAX/Ramtek graphics into any application package. These macro-routines will perform the necessary FLIK calls to execute the desired graphics effect. The following paragraphs discuss this macro-routines package subdivided by effect.

c. On first glance, these routines may seem simple (that's good) and/or restrictive (that's a function of keeping it simple) but the author is certainly open to, and appreciative of, ideas that might improve the utility of the package.

2. Preparation. To use the Macro-FLIK package the applications programmer must link his application software appropriately. System logicals (automatically assigned at your login) ØMACRO, ØKGL, ØRMSTN, and ØMARQ, will point your link to the appropriate directories. A sample link follows:

```
     LINK      APPLICATION, ØMACRO:MACRO/OPT
```

Where MACRO.OPT is as follows:

```
     ØMACRO:A/LIB           (The Macro-FLIK library)
     ØKGL:A/LIB             (The FLIK library)
     ØRMSTN:A/LIB           (The Ramtek station/chassis info lib)
     ØMARQ:QIO_MARQ/LIB     (The vendor-provided Ramtek Marquis
                             driver software library)
```

3. Initialization. Before communication can be established with one of our Ramtek color monitors, the user must execute a series of initialization routines. These range from opening and assigning a channel to the Ramtek controller and associating the appropriate monitor and graph tablet with your workstation, to defining and loading colors into the Ramtek video lookup table (VLT) for later reference. Macro-FLIK *simplifies* this procedure by requiring the application programmer to call only one routine, one time, and with no calling arguments.

a. INIT_RAMTEK. Calling format:

CALL INIT_RAMTEK
This routine performs all of the necessary initialization to establish the link between user terminal and graphics work station based on several interactive queries of the user. The queries (in bold type), typical response, and discussion of possible responses are shown below.

ENTER RAMTEK LOGICAL SUCH AS RMA1:, OR END RMAØ:

The user is expected to enter the four-character logical associated with the particular work station (there is a label on each workstation at TRAC-FLVN Central Computer Facility (CCF) and Wargame Computer Facility (WCF)).

RELOAD STATUS FROM A PREVIOUS RUN? N

Appropriate responses are Yes or No. This option allows the user to reload the status of curve displays for review/modification. Typically the user's response will be No. For more information, see paragraph 6a.

BUILD A NEW COLOR FILE (N)?
USE DEFAULT COLOR FILE (D)?
USE OLD, PREVIOUSLY CREATED COLOR FILE (O)? N

The first time this software is executed, the user will likely select responses New or Default. Typically thereafter, the Old option would be selected to use a previously created and saved color file. The prompts which follow are a result of having selected the New option. Had the user selected Default, the next prompt to appear would be the "SAVE THE COLOR FILE JUST CREATED?" prompt (discussed below). Had the user selected Old, he would be prompted to enter the filename of that old color file, followed by the prompt "DO YOU WANT TO SEE YOUR COLOR SCHEME?" (discussed below).

(1) Building a color file. Prior to continuing the discussion of remaining prompts and responses, it's necessary to provide the user with at least a cursory understanding of color file production.

The Maro-FLIK color scheme generation software is more sophisticated than it needs to be. For the typical application, the user should simply specify one buffer and build an ample number (current software restriction is 32) of colors by responding to the prompts that follow. However, the capability exists for the educated user to devise a sophisticated overlay scheme. That procedure is described in more detail in the following paragraphs.

Current TRAC Ramtek hardware is configured with anywhere between eight and sixteen usable refresh memory planes per station which will allow addressing a maximum of 256 (2**8) to 65,536 (2**16) colors loaded into the video lookup table (VLT). By cleverly loading the VLT and sacrificing colors, an overlaying effect can be achieved. This clever loading (the details of which are beyond the scope of this manual) is accomplished for the user by a FLIK routine which requires that the user define the color scheme in terms of overlay buffers and colors within each buffer. A buffer consists of a number of refresh memory planes where each plane provides an additional power of two-color capacity. This means that a buffer consisting of three planes would allow the user to load eight colors (2**3). Of course, the advantage to such a buffering/overlay scheme is that graphics drawn in any buffer can be "hidden from view" by drawings in a higher number buffer, but become visible again once the higher buffer drawings are erased.

Currently, the Macro-FLIK software restricts the user to eight buffers using as many planes as are available at his current work station (but, as mentioned previously, limited to a maximum of five planes per any one buffer). Of course, a single buffer would provide no overlays, simply the generation of thirty-two colors (five planes or 2**5 colors). The responses made to the sample prompts below would create an overlay scheme consisting of

3 overlay buffers of 32, 2, and 4 colors (5, 1, and 2 planes) respectively.

If the user plans to incorporate pull-down/pop-up menus (see paragraph 7), the software will generate a temporary top level "artificial" buffer of 2 planes (if available at the current station; otherwise, the user is warned that graphics drawn in his top buffer may be "overdrawn" if he uses pull-down/pop-up menus). It is in this temporary buffer that these special menus are drawn, thus allowing them to overlay the user's graphics work without (except in the case just mentioned) ill effect. The pull-down/pop-up grid, text, and highlight colors respectively will be duplicates of the first three colors (excluding the "clear" color number 1) in the user-generated top buffer.

For example: If the user, in responding to the sample prompts below, selected colors red, green, and blue as his three colors in buffer number 3 (his top buffer), and then chose to use pull-down menus in his software, the menus would appear with green text on a red background (grid) and upon selection would be highlighted in blue (not a pretty sight). If the user's current work station has more than the eight planes used in this color file definition available, a temporary buffer 4 will be generated with those three colors. If, however, this station has only eight planes available, then the pull-downs/pop-ups will be drawn in buffer number 3, effectively erasing the user's graphics drawn in that buffer. Although the software doesn't prohibit this type of potential damage, it does warn the user at initialization.

The following prompts actually appear on the Ramtek monitor, and, consequently, the user's responses are entered via the graph tablet and puck. For this reason, I've chosen to show the response(s) in < >.

SELECT NUMBER OF OVERLAY BUFFERS <3>

Hardware limit (practically speaking) is 8. Entering 1 is acceptable, but implies no "overlaying."

SELECT # OF PLANES IN OVERLAY BUFFER #1 <5>

This prompt will obviously appear for each buffer. Current software limit is 32 colors (five planes as in this example) per buffer. As discussed in para 3a(3), the user is limited to the number of planes available at his current station and a total of eight buffers.

SELECT # OF PLANES IN OVERLAY BUFFER #2 <1>

SELECT # OF PLANES IN OVERLAY BUFFER #3 <2>

Note the total number of planes for all buffers is at most (in this case, exactly) eight.

OVERLAY BUFFER CONFIGURATION
SELECT COLORS USING THE TABLET

An empty matrix representing the user-specified color scheme is displayed at this time, along with a color pallet of available colors from which the user may select. An "X" appears in each matrix color box and is replaced by the color selected from the pallet as the user progresses through his color scheme definition.

DO YOU WISH TO MAKE ANY CHANGES? <Y>

Appropriate "selections" are Yes or No. If the user selects No, the next prompt to appear will be the "SAVE THE COLOR FILE..." prompt.

SELECT COLOR TO CHANGE

The user selects, via the graph tablet and puck, the box in his (now filled) color matrix which contains the color to be changed.

SELECT NEW COLOR FROM PALLET

The user similarly selects from the color pallet the replacement color. These two prompts are repeated until the "key when finished" box is selected.

SAVE THE COLOR FILE JUST CREATED? Y

ENTER COLOR FILE NAME (NO EXTENSION) COLORFILE

START OVER? N

Appropriate responses are Yes or No. If the user selects Yes, this option allows him to build another color file by clearing his color matrix and recycling starting with the "SELECT NUMBER OF OVERLAY BUFFERS" prompt.

DO YOU WANT TO SEE YOUR COLOR SCHEME? N

Appropriate responses are Yes or No. If the user selects Yes, his color scheme is listed to the CRT.

IS THIS COLOR SCHEME WHAT YOU WANT? Y

Appropriate responses are Yes or No. If the user selects No, he will return to the "BUILD A NEW..." prompt.

4. Graphics input. Input data displayed in menus on color monitors and selected via the workstation graphics tablet by

light pen/puck was the first application addressed. Since nearly all application software requires user interface, this seemed a high payoff undertaking. The assumption made here is that graphically inputting data is better than keyboard entry because of its aesthetic appeal, user friendliness, and/or simplicity. An application programmer who accepts this assumption would then prefer to incorporate graphics input in his application package if it was not too burdensome to effect and was satisfactorily responsive. This three-routine menu package attempts to provide the user that capability in such a manner. A more sophisticated menuing package, featuring pull-down and pop-up menus, is discussed in paragraph 7.

a. DRAW_MENU   Calling format:

```
CALL DRAW_MENU (LOC_MENU, ERASE, MEN_BUFF, GRID_COLOR,
                TEXT_COLOR, HILITE_COLOR, NUM_BOXES, TEXT)
```

To keep it simple, this software offers the user four possible locations for menu display. He may have a menu displayed in each location simultaneously or choose to display menus only one at a time. The first calling argument, LOC_MENU, is a numeric code 1 through 4 identifying the desired location of the menu in a clockwise manner with 1 being top of screen (1=top, 2=right, etc.). The next argument is a logical variable indicating if the user wants this menu to be automatically erased once a selection is made. The next four arguments are interconnected. MEN_BUFF indicates the graphics overlay buffer (user defined by his responses to routine INIT_RAMTEK described in paragraph 3(a)) and the next three arguments indicate the corresponding colors in which the menu will display. NUM_BOXES notifies the software of how many entries the menu is to contain, and TEXT is a character array containing the actual text to appear in the menu. (User note: the menu size is dynamically determined so lengthy text should be reserved for the side menus). This and other calls to routines in this menu package may be clearer by looking at the sample application program in paragraph 4(d).

b. MONI_MENU   Calling format:

```
CALL MONI_MENU ( MEN1, MEN2, MEN3, MEN4, MEN_OUT, MEN_BOX)
```

This routine is called to monitor (i.e., await graph tablet selection from) any combination of the menus previously displayed via DRAW_MENU. The user simply identifies which menu(s) is/are to be monitored by the first four arguments (1 indicating the menu is to be monitored and Ø if not). The software will then wait for the user to input data via the graph tablet (i.e., make a selection using the tablet). The arguments MEN_OUT, and MEN_BOX return selection information. MEN_OUT is the code number representing the location of the menu from which the selection was made and MEN_BOX the corresponding box number within that menu.

c. ERAS_MENU Calling format:

CALL ERAS_MENU (MENU_NUMBER)

If the user has menus to be displayed from which multiple selections will be made, he will not want the menu to erase after each selection. By setting the erase "flag" accordingly when the menu is drawn this will not occur. However, a means of eventually erasing this menu may still be desirable. ERAS_MENU allows the user to erase any or all menu(s) by passing the location code number.

d. Menuing application example.

NOTE: In this example, remember the first argument in the DRAW_MENU call represents the menu location, and only <u>coincidentally</u> the menu number.

```
 PROGRAM APPLICATION
C      ************************************************************
C      * THIS PROGRAM IS DESIGNED MERELY TO ILLUSTRATE SIMPLE
C      * MENU DISPLAYS AND SUBSEQUENT MONITORING.  THE PROGRAM
C      * DRAWS MENU # 1 AT THE TOP OF THE SCREEN AND, BASED ON
C      * THE USER'S SELECTION FROM THAT MENU DRAWS, MENU 2 (AT
C      * THE RIGHT) OR MENU 3 (AT BOTTOM) FOR SUBSEQUENT SELEC-
C      * TION.  ONE ADDITIONAL MENU, MENU 4 (AT LEFT), IS GENER-
C      * ATED AFTER SELECTION FROM MENU 2.

       CHARACTER*20  TXT1(3), TXT2(8), TXT3(4), TXT4(20)
       LOGICAL ERAS

C      *************** DEFINE 4 MENUS ****************
C      *MENU 1
       DATA TXT1 / 'SELECT SYSTEM TYPE', 'PROCESS DATA', 'END
       THE PROGRAM'/

C      *MENU 2
       TXT2(1) = 'FIXED WING AIRCRAFT'     ! NOTE LONGER TEXT
       TXT2(2) = 'ROTARY WING AIRCRAFT'    ! WORKS BEST IN SIDE
       TXT2(3) = 'SP ARTILLERY'            ! MENUS
       TXT2(4) = 'TOWED ARTILLERY'
       TXT2(5) = 'TANKS'
       TXT2(6) = 'ARMORED PERS CARRIERS'    ! NOTE 21ST CHAR TRUN.
       TXT2(7) = 'LOGISTICS'
       TXT2(8) = 'TRUCKS'

C      *MENU 3
       TXT3(1) = 'COMPUTE MEAN'
       TXT3(2) = 'COMPUTE STAND DEV'
       TXT3(3) = 'COMPUTE RANGE'
       TXT3(4) = 'COMPUTE VARIANCE'

C      *MENU 4
```

```
      DO I = 1,20
         WRITE (TXT4(I), '(I2)') I          ! LOAD #'S THEMSELVES
      ENDDO

C     ***************** MENU DEFINITIONS COMPLETE ***************

C    *INITIALIZE RAMTEK (OPEN CHANNEL TO APPROPRIATE STATION,
C    *ESTABLISH COLOR SCHEME TO BE EMPLOYED BY OTHER CALLS)
     CALL INIT_RAMTEK                          ! ONE TIME CALL

C    *DRAW TOP MENU
C    * CALLING ARGUMENTS (INPUT) ARE AS FOLLOWS
     LOC = 1             ! LOCATES THIS MENU AT TOP
     ERAS = .FALSE.      ! MENU WILL NOT ERASE AFTER SELECTION
C    *THE COLORS REPRESENTED BY THE FOLLOWING ARGUMENTS ARE
C    *DEPENDENT UPON THE USER'S RESPONSES TO INIT_RAMTEK PROMPTS
     MENBUF = 1          ! OVERLAY BUFFER
     MGRID = 2           ! MENU GRID COLOR (USER DEFINED BUFFER 1,
                         ! COLOR 2
     MTXT = 3            ! MENU TEXT COLOR
     MHLIT = 4           ! COLOR IN WHICH MENU SELECTION IS
                         ! HILITED
     MBOX = 3            ! NUMBER OF ENTRIES IN THIS MENU
     CALL DRAW_MENU (LOC, ERAS, MENBUF, MGRID, MTXT, MHLIT,
    *                MBOX, TXT1)
 100 CONTINUE

C    *MONITOR ONLY THE TOP MENU (AS INDICATED BY THE 1 IN ONLY
C    *THE FIRST OF THE FOUR AVAILABLE MENU LOCATIONS SLOTS)
     CALL MONI_MENU (1, 0, 0, 0, MENOUT, MENBOX)

     IF (MENBOX .EQ. 1) THEN          ! "SELECT SYSTEM TYPE"
        CALL DRAW_MENU (2, .TRUE., 2, 5, 4, 3, 8, TXT2)
     ELSEIF (MENBOX .EQ. 2) THEN      ! "PROCESS DATA"
        CALL DRAW_MENU (3, .TRUE., 1, 7, 5, 4, 3, TXT3)
     ELSE                             ! "END PROGRAM"
        GO TO 9999
     ENDIF

C    *MONITOR BOTH OF THE DISPLAYED MENUS
     CALL MONI_MENU (0, 1, 1, 0, MENOUT, MENBOX)

     IF (MENOUT .EQ. 2) THEN           ! SELECTION FROM RIGHT MENU
        PRINT*,'HOW MANY OF THIS SYSTEM TO PROCESS?'
        CALL DRAW_MENU (4, .TRUE., 1, 7, 3, 4, 20, TXT4)
        CALL MONI_MENU (0, 0, 0, 1, MENOUT, MENBOX)
        PRINT*,'PROCESSING ',MENBOX,' SYSTEMS'
     ELSE                              ! SELECTION FROM BOTTOM MENU
        PRINT*,'COMPUTING STATS'
     ENDIF

     GO TO 100

 9999 CONTINUE
```

```
      CALL ERAS_MENU (1)      ! NOTE CALLING WITH Ø ERASES ALL
                              ! MENUS STILL DISPLAYED
      STOP 'END OF APPLICATION'
      END
```

5. Point data display. Another graphics application that is considered to be relatively high payoff is the ability to graphically display and link point data. This capability has application both to output and input data manipulation. Giving the user dynamic, graphically accomplished, curve-modification capability provides a flexible, user friendly means of developing data files. Allowing the user to display data graphically (with the eventual ability to produce hard copies) could be especially useful in analyzing the data. To further expand the latter capability, graph "fills" are available allowing the user to generate cumulative distribution type displays. As in the previous section, an application example is provided in section 5e.

a. LOAD_CURV   Calling format:

```
CALL LOAD_CURV (CURV_BUF, CURV_CLR, X, Y)
```

This routine is called to load a curve for later display. Currently, software limits the user to 2Ø curves of 3Ø or less points each. The user controls the overlay buffer, CURV_BUF (1 if buffering is not being used) and color, CURV_CLR, of the curve as determined by his repsonses to the INIT_RAMTEK prompts. The arrays X and Y define the X and Y-coordinates respectively of the data points in the order they are to be plotted. (NOTE: Macro-FLIK software requires that these points be ordered in ascending order of X.)

b. DRAW_CURV   Calling format:

```
CALL DRAW_CURV (AXES_BUF, AXES_CLR, FILL, ACCUM)
```

This routine will display **all** curves previously loaded via LOAD_CURV and not subsequently erased from memory via ERAS_CURV. The axes are dynamically determined with the user having the option to identify each axis' increment if he chooses. The user controls the buffer and color of the axes by the first two arguments in this call. The argument FILL is a logical which, when .TRUE., will appropriately color fill "below" each curve after plotting it. Otherwise only the curves themselves are plotted. The argument ACCUM will produce a cumulative distribution type of display by summing the Y-coordinates at each increment point. If FILL is .FALSE., the display will plot all curves and the "cumulative" curve unless the data is "nice" (all X coordinates the same). In that case the first curve will be drawn and each subsequent curve will reflect the sum of its Y-coordinates with those of the previous curves. If FILL is .TRUE. the user will get an error message unless, as before, the data is "nice," in which case he will get the same set of cumulative curves mentioned above with appropriate color fills.

EXAMPLE: The user loads (via LOAD_CURV) 2 curves defined as follows:

```
        CURVE 1 - (Ø,Ø),  (1Ø,4Ø), (3Ø,3Ø), (6Ø,8Ø)
        CURVE 2 - (Ø,1Ø), (1Ø,2Ø), (3Ø,5Ø), (6Ø,1ØØ)
```

First of all, the axes will be dynamically computed with origin (in this case) at Ø,Ø and X-axis of length 6Ø (X range) and Y-axis of length 1ØØ (Y range). Increments will default to 6 units on the X-axis (1Ø equal increments) and 1Ø on the Y-axis, or the user may specify increments of his choice. Next, the points will be plotted and connected forming the two curves defined (in the colors previously assigned via LOAD_CURV). What occurs next depends on the value of the final two calling arguments as follows:

CASE 1: FILL = .FALSE. ACCUM = .FALSE.
Nothing else to do. What you see is what you get.

CASE 2: FILL = .TRUE. ACCUM = .FALSE.
Curve 2 would display first (has the highest Y) and immediately fill below, followed by curve 1 with corresponding fill.

CASE 3A: FILL = .FALSE. ACCUM = .TRUE. "NICE DATA"
Note: NICE DATA means the X-coordinates of all curves are the same as in our example Ø, 1Ø, 3Ø, 6Ø.
All curves will "erase" (since new axes must be computed) and then redraw on the new axes. The first curve will be drawn followed by a second which is, in fact, the accumulation of the first and the second.

CASE 3B: FILL = .FALSE. ACCUM = .TRUE. "NOT NICE DATA"
As in case 3A, all curves will erase. This time they will reappear one at a time and one additional curve (which represents the accumulation of all the others) will appear in the "axes color and buffer."

CASE 4: FILL = .TRUE ACCUM = .TRUE.
In the case where the data is "nice," this display will be the same as case 3B except the appropriate color fills will indicate the portion of the accumulation represented by each curve. If the data is "not nice," the software will issue a warning that a filled cumulative display is impossible and would be meaningless and subsequently produce a nonfilled (a la 3B) display.

c. MONI_CURV Calling format:

CALL MONI_CURV (NUMCURV)

MONI_CURV is a multipurpose routine. On one hand, the user may call this routine to dynamically modify (and subsequently save) the point data, or he may simply wish to attain statistical information regarding the curve(s) being monitored.

By passing NUMCURV as Ø (typically the easiest), all currently displayed curves can be monitored; otherwise, only the curve specified by the calling argument value will be monitored. Note that the numbering sequence is the user's responsibility with the curves numbered sequentially as they are "loaded" and packed (see section 5d) as they are "erased" from memory.

If the user is monitoring all curves, he is prompted to graphically select one for more detailed inspection. The curve selected (either by the action just described or by specific reference as the calling argument) will then highlight its current data points (nodes). The routine will next display a menu at the top of the Ramtek monitor querying the user for the type of monitoring of interest. The user may choose one of the following:

(1) Move a node. This option allows the user to relocate a data point which, in turn, modifies the point data in memory for that curve.

(2) Delete a node. This option allows the user to erase a data point from the screen, reconnect the preceding and succeeding points, and subsequently erase that data point in memory for that curve.

(3) Add a node. This option allows the user to expand the curve by adding a new data point to the end of the curve (far right). A word of caution here, if the user wishes to expand the curve which extends farthest to the right of all curves displayed, it will be necessary to "create a dummy curve" since the axes are dynamically determined based on the minimums and maximums of the curve(s) to be displayed.

(4) Insert a node. This option allows the user to place a new data point on the curve between two existing data points, with the corresponding effect on the curve in memory.

(5) Stats. This option allows the user to obtain statistical information regarding the selected curve. Information such as range, mean, and standard deviation in both the X and Y directions is available. Also available is point selection information, where the user can identify any point on the curve (or off) by simply locating the light pen/puck and depressing it. The information displayed identifies the exact location selected as well as the point on the curve (perpendicularly) nearest that selected.

d. ERAS_CURV   Calling format:

```
CALL ERAS_CURV ( NUMBER_CURV, MEMORY )
```

This routine allows the user to erase a specified curve or all currently displayed curves (NUMBER_CURV = Ø) from the screen and optionally from memory. If the user erases the curve(s) from memory, the remaining curves will be packed (i.e., the third curve originally loaded would move to second if either of the first two were erased). WARNING: remember it's the user's obligation to handle the curve ordering.

e. Data point display/monitor example.

```
         PROGRAM APPLICATION2
C       ***************************************************************
C       * THIS ROUTINE ILLUSTRATES DEFINING, LOADING, DISPLAYING,
C       * AND MONITORING CURVES.  FIRST THE TANK, LAW, AND HELO
C       * CURVES ARE LOADED, DISPLAYED, AND MONITORED.  NEXT, THE
C       * LAW CURVE IS ERASED, AND THE OTHERS ARE REDRAWN WITH
C       * THE FILL OPTION ACTIVATED (AND PAUSES FOR REVIEW).
C       * NEXT, THE REMAINING CURVES ARE ERASED FROM THE SCREEN,
C       * A NEW CURVE (ARTY) IS LOADED AND ALL ARE REDRAWN WITH
C       * BOTH FILL AND ACCUMULATE OPTIONS SELECTED (PAUSE).
C       * FINALLY, THE TANK AND HELO CURVES ARE ELIMINATED AND
C       * THE ARTY CURVE IS DRAWN ALONE FOR MONITORING.

         REAL HOUR(7),TANK(7), LAW(7), HELO(7), ARTY(7)
         LOGICAL FILL, ACCU

C       *DATA MAY BE ROUTINE GENERATED OF COURSE

C       *DEFINE "Y-COORDINATES" OF ALL DATA POINTS
         DATA TANK /3.1, 2.6, 8.Ø, 5.5, 3.1, 1.3,  .4/
         DATA LAW  /1.4, 1.6, 4.2, 2.2,  .8,  .2, Ø. /
         DATA HELO /Ø. , 2.1, 9.8, 6.7, 4.Ø, Ø. ,  .1/
         DATA ARTY /6.2, 8.9,16.2, 1Ø.1,6.4, 3.5, 1.1/

C       *DEFINE "X-COORDINATES" OF ALL DATA POINTS
         DATA HOUR / 1., 2., 3., 4., 5., 6., 7. /

         CALL INIT_RAMTEK

C       *THE COLORS REPRESENTED BY THE FOLLOWING ARGUMENTS ARE
C       *DEPENDENT UPON THE USER'S RESPONSES TO INIT_RAMTEK
         NBUF = 1     ! OVERLAY BUFFER IN WHICH CURVE WILL DRAW
         NCLR = 5     ! 5TH COLOR IN BUFFER 1, USED FOR TANK CUR
         NPTS = 7     ! NUMBER OF DATA POINTS IN THIS CURVE

         CALL LOAD_CURV (NBUF, NCLR, NPTS, HOUR, TANK)    !TANK

         CALL LOAD_CURV (NBUF,    3, NPTS, HOUR, LAW)     !LAW

         CALL LOAD_CURV (NBUF,    4, NPTS, HOUR, HELO)    !HELO

C       *DRAW THE 3 CURVES JUST LOADED WITH NO FILL NOR ACCUM.
         NBUF = 1     ! OVERLAY BUFFER IN WHICH AXES WILL DRAW
```

```
          NCLR = 6     ! 6TH COLOR IN BUFFER 1, USED FOR AXES
          FILL = .FALSE.    ! DO NOT FILL UNDER THE CURVES
          ACCU = .FALSE.    ! DO NOT PRODUCE A CUMULATIVE CURVE

          CALL DRAW_CURV (NBUF, NCLR, FILL, ACCU)

C        *MONITOR ALL THREE CURVES FOR ADDITIONAL INFORMATION OR
C        *MODIFICATION AND SAVE TO FILE IF DESIRED (SAVE_STAT)
          CALL MONI_CURV (Ø)

C        *ERASE THE LAW CURVE, REDRAW THE OTHERS WITH FILL
          CALL ERAS_CURV (2, 1)       !NOTE CURVE ERASED FROM
                                      !BOTH SCREEN & MEMORY
          CALL DRAW_CURV (1, 6, .TRUE., .FALSE.)

          PRINT*,'HIT ANY KEY TO CONTINUE'    !PAUSE TO EXAMINE
          READ(5,G) JUNK

          CALL ERAS_SCRN (Ø,Ø)       !ERASE SCREEN, ALL BUFFERS

          CALL LOAD_CURV (1, 5, 7, HOUR, ARTY)    !LOAD ARTY CURVE

          CALL DRAW_CURV (1, 6, .TRUE., .TRUE.)

          PRINT*,'HIT ANY KEY TO CONTINUE'    !PAUSE TO EXAMINE
          READ(5,G) JUNK

          CALL ERAS_SCRN (Ø,Ø)        !ERASE SCREEN, ALL BUFFERS
          CALL ERAS_CURV (1, 1)       !ERASE TANK CURVE FROM MEMORY
          CALL ERAS_CURV (2, 1)       !ERASE HELO CURVE (NOTE DATA
                                      !WAS PACKED)

          CALL MONI_CURV (3)          !MONITOR ARTY CURVE

          STOP
          END
```

6. <u>General screen and status preservation.</u> The capabilities described in the previous section allow for dynamic display and modification of curve data. Dynamic display has obvious applicability for analysis, especially when the stats option is selected while monitoring the curve(s) of interest. Dynamic modification, on the other hand, is of little use unless the "picture" and/or overall "system status" can be preserved for later use. Three of the MACRO-FLIK callable routines discussed in this section provide the application programmer with this capability, while the other provides a means for clearing the monitor screen completely. Several sample application programs (sections 6f,g,h) are provided for further assistance.

a. SAVE_STAT   Calling format:

```
CALL SAVE_STAT
```

This routine can be called directly by the user, as in section 6f, but more likely will be called by the software from routine MONI_CURV at the user's request. In other words, when the user has completed monitoring displayed curves, the option to preserve the current status (which includes color scheme, screen data, and all curve data) is proposed by the software itself. Status files are transportable between low and high resolution sytems.

b. LOAD_STAT   Calling format

```
CALL LOAD_STAT (VALID)
```

This routine is use to reload the status (color scheme, screen data, and all curve data) saved during a previous run via SAVE_STAT. The returned logical calling argument, VALID, simply notifies the user that a valid status file (.STAT) was found.

c. SAVE_PIC   Calling format:

```
CALL SAVE_PIC
```

This routine saves only the picture currently displayed on the Ramtek monitor. (There is no preservation of curve data or color scheme as with SAVE_STAT). Currently picture files are one-way-transportable between low and high resolution monitors. this means that pictures generated and saved on a low resolution system can be displayed (via LOAD_PIC) on a high resolution system with no ill effects. However, only the lower left quarter of pictures generated and saved on a high resolution system will display on low resolution screen.

d. LOAD_PIC   Calling format:

```
CALL LOAD_PIC
```

This routine loads a picture previously preserved with SAVE_PIC. There is no preservation of color when saving a picture, so it's the user's responsibility to assure that an appropriate color scheme is loaded prior to loading the picture.

e. ERAS_SCRN   Calling format:

```
CALL ERAS_SCRN (MEMORY, BUFFER)
```

This routine allows the user a means of completely clearing the monitor screen by erasing any or all (BUFFER=Ø) buffers. The first argument, MEMORY, will typically be Ø, but can be set to 1 if the user wishes to clear memory of all previously loaded curve data (to freshly load new curve data).

f. General screen and status application, example 1.

```
    PROGRAM APPL_LOAD
```

```
C       *THIS PROGRAM SIMPLY LOADS SOME CURVES & SAVES THEM
         REAL HOUR(7),TANK(7), LAW(7), HELO(7), ARTY(7)

C       *DATA MAY BE ROUTINE GENERATED OF COURSE

         DATA TANK /3.1, 2.6, 8.Ø, 5.5, 3.1, 1.3, .4/
         DATA LAW /1.4, 1.6, 4.2, 2.2, .8, .2, Ø.  /
         DATA HELO /Ø.  , 2.1, 9.8, 6.7, 4.Ø, Ø.  , .1/
         DATA ARTY /6.2, 8.9,16.2, 1Ø.1,6.4, 3.5, 1.1/

         DATA HOUR / 1., 2., 3., 4., 5., 6., 7.  /

         CALL INIT_RAMTEK

         CALL LOAD_CURV (1, 5, 7, HOUR, TANK) !LOAD TANK CURVE

         CALL LOAD_CURV (1, 3, 7, HOUR, LAW) !LOAD LAW CURVE

         CALL LOAD_CURV (1, 4, 7, HOUR, HELO) !LOAD HELO CURVE

         CALL SAVE_STAT        ! SAVE STATUS FOR LATER UPDATE

         STOP
         END
```

g. General screen and status application, example 2.

```
         PROGRAM APPL_MODIFY
C       *THIS ROUTINE ALLOWS THE USER TO DISPLAY, ANALYZE,
C       *MODIFY, AND SUBSEQUENTLY SAVE CURVES LOADED BY APPL_LOAD
C       *AND THEN CONTINUE MODIFYING THOSE CURVES IF DESIRED

         CHARACTER RESP*1 /'Y'/

         CALL INIT_RAMTEK        !SAY YES TO RELOAD PROMPT
                                 ! AND ENTER NAME OF STATUS FILE
                                 ! SAVED IN APPL_LOAD

         DO WHILE (RESP .EQ. 'Y')
            CALL DRAW_CURV (1, 8, .FALSE., .FALSE.)
            CALL MONI_CURV (Ø)    !OPTIONALLY SAVE STATUS
            CALL SAVE_PIC         !OPTIONALLY SAVE PICTURE
            CALL ERAS_SCRN (Ø,Ø)  !ERASE ALL BUFFERS, BUT NOT
                                  !CURVE DATA
            PRINT*,'CONTINUE?'
            READ(5,1Ø) RESP
    1Ø      FORMAT (A1)
         ENDDO

         STOP
         END
```

h. General screen and status application, example 3.

```
       PROGRAM APPL_RELOAD
C      *THIS ROUTINE ALLOWS A USER TO LOAD PREVIOUSLY PRESERVED
C      *STATUS OR PICTURE FILES.

C      *WARNING: REMEMBER PICTURE FILES ARE NOT TRANSPORTABLE

       CHARACTER RESP*1

       CALL INIT_RAMTEK

    1Ø PRINT*,'LOAD STATUS (S), OR PICTURE (P)?'
       READ(5,2Ø) RESP
    2Ø FORMAT (A1)

       IF (RESP .EQ. 'S') THEN
          CALL LOAD_STAT (VALID)
          CALL DRAW_CURV (1, 8, .FALSE., .FALSE.)
          CALL MONI_CURV (Ø)
          CALL SAVE_PIC
       ELSEIF (RESP .EQ. 'P') THEN
          CALL LOAD_PIC
       ELSE
          GO TO 9999
       ENDIF

       PRINT*,'ERASE SCREEN?'
       READ(5,2Ø) RESP
       IF (RESP .EQ. 'Y') CALL ERAS_SCRN (Ø,Ø)

       GO TO 1Ø
  9999 STOP
       END
```

7. Pull-down and pop-up menuing. Paragraph 4 discusses Macro-FLIK's initial, generic, and simplistic approach to menuing. Pull-down and pop-up menus provide significantly more aesthetic appeal, but they aren't an appropriate formats for every menuing application. A pull-down menu consists of a main menu, which appears at the top of the monitor, and a set (currently software restricted to 8 or less) of submenus, which visually "pull down" from the main menu as its entries are "touched" via the graph tablet pen/puck. A pop-up menu is one which appears in the center of the graphics monitor for a single selection and automatically disappears after that selection is made. Again, an application program to illustrate pull-down and pop-up implementation is provided in paragraph 7(d). The nature of these menus (they overlay previous displays) mandates that they be drawn in the highest overlay buffer defined by the user in his responses to routine INIT_RAMTEK (paragraph 3(a)), or in a temporary, artificial, software generated higher buffer so as not to disturb the underlying graphics. The first three (excluding "clear" color 1) colors in that buffer are reserved to define the

menu grid, text, and hilight colors.

a. LOAD_PULL   Calling format:

```
CALL LOAD_PULL (NUM_BOX_MAIN, TXT_MAIN, MAX_SUB, NUM_BOX_SUB,
                TXT_SUB)
```

This routine loads the user-defined pull-down menu text into Macro-FLIK commons for later display and monitoring. Typically, the user will define the menu text and call LOAD_PULL from one routine and then display and monitor the menu as needed. The first two calling arguments refer to the main menu portion of the pull-down with NUM_BOX_MAIN being the number of entries (boxes) in the main menu (note the above restriction of 8) and TXT_MAIN is the character array containing the text for each of these entries. The last three arguments describe the submenu portion. NUM_BOX_SUB is an array indicating the number of entries in each submenu. For example: if there are three main menu entries, there would most likely be three corresponding submenus. The submenus can each have a different number of entries, say 3, 5, and 7. In this case, the array NUM_BOX_SUB would be dimensioned 3 and defined as follows: NUM_BOX_SUB(1) = 3, NUM_BOX_SUB(2) = 5 and NUM_BOX_SUB(3) = 7. MAX_SUB represents the largest of all the NUM_BOX_SUB values (7 in this example), and TXT_SUB is a two-dimensional character array containing the submenu text.

b. MONI_PULL   Calling format:

```
CALL MONI_PULL (WHICH_SUB, WHICH_BOX_IN_SUB)
```

This routine displays the pull-down menu previously loaded into common via LOAD_PULL and monitors it in the following fashion. As the user slides the pen/puck (no button depression required) into a main menu entry box, the appropriate submenu will appear below that main menu entry (the submenu text length is not restricted by that of the main menu entry). As the user slides the pen/puck through the submenu, each entry will highlight to clearly identify which would be selected should the user depress the puck button (#1 on a four-button puck). The user's selection will highlight and the submenu subsequently disappear, however, the main menu remains displayed to allow for successive monitoring until the user erases it via ERAS_SCRN (paragraph 6e). The selection information is returned to the calling routine through the calling arguments as follows: the number of the displayed submenu (left to right) from which the selection was made is returned in WHICH_SUB, while the number (top to bottom) of the selected entry box in that submenu is returned in WHICH_BOX_IN_SUB.

c. MONI_POP   Calling format:

```
CALL MONI_POP (NUMBOXES, TEXT, WHICH_BOX_IN_POP)
```

Since a pop-up menu appears only long enough for a single selection, this routine both displays and monitors the user-defined pop-up menu. The user provides the number of entries (boxes) for the pop-up in argument NUMBOXES and a text array containing the pop-up text in TEXT. WHICH_BOX_IN_POP is returned to the user as the number (top to bottom) of the selected entry.

d. Pull-down/pop-up menuing example program.

```
      PROGRAM APPL_POP
C    ***************************************************
C    * THIS ROUTINE DISPLAYS AND ALLOWS CYCLICAL SELECTION
C    * FROM A PULL-DOWN MENU UNTIL THE END PROGRAM SUBMENU
C    * ENTRY IS SELECTED.  ADDITIONALLY, IF THE USER SELECTS
C    * THE EDIT SUBMENU AND SUBSEQUENTLY THE DELETE OPTION,
C    * A POP-UP MENU REQUESTING CONFIRMATION WILL APPEAR.

      CHARACTER MTXT(4)*7, STXT(6,4)*30, PTXT(3)*15
      INTEGER NSUB(4)

C    ***** DEFINE THE PULL-DOWN MENU *****
C    * MAIN MENU PORTION
      MTXT(1) = 'SELECT'
      MTXT(2) = 'PROCESS'
      MTXT(3) = 'NUMBER'
      MTXT(4) = 'EDIT'
      NMAIN = 4

C    * SUBMENU PORTION
      STXT(1,1) = 'FIXED-WING AIRCRAFT'
      STXT(2,1) = 'ROTARY-WING AIRCRAFT'
      STXT(3,1) = 'SP ARTILLERY'
      STXT(4,1) = 'TOWED ARTILLERY'
      STXT(5,1) = 'TANKS'
      STXT(6,1) = 'ARMORED PERSONNEL CARRIER'
      NSUB(1) = 6

      STXT(1,2) = 'COMPUTE MEAN'
      STXT(2,2) = 'COMPUTE STANDARD DEVIATION'
      STXT(3,2) = 'COMPUTE RANGE'
      STXT(4,2) = 'COMPUTE VARIANCE'
      NSUB(2) = 4

      DO I = 1, 20
         WRITE ( STXT(I,3),'(I2)') I
      ENDDO
      NSUB(3) = 20

      STXT(1,4) = 'ADD'
      STXT(2,4) = 'CHANGE'
      STXT(3,4) = 'DELETE'
      STXT(4,4) = 'DUPLICATE'
      STXT(5,4) = 'END PROGRAM'
      NSUB(4) = 5
```

```
          MAXSUB = 20

C        ***** BUILD POP UP MENU *****
          PTXT(1) = 'CONFIRM'
          PTXT(2) = 'REJECT'
          PTXT(3) = 'NONE OF THE ABOVE'

C        ***** FINISHED DEFINING MENUS *****

C        *LOAD THE PULL DOWN MENU INTO COMMON
          CALL LOAD_PULL (NMAIN, MTXT, MAXSUB, NSUB, STXT)

C        *DISPLAY AND MONITOR THE PULL DOWN MENU
   100    CALL MONI_PULL (NSUB, NBOX)

C        *IF USER SELECTS "END PROGRAM"
          IF (NSUB .EQ. 4 .AND. NBOX .EQ. 5) THEN
             CALL ERAS_SCRN (0,0)
             STOP 'END OF APPL_POP'
          ENDIF

C        *IF USER SELECTS EDIT/DELETE
          IF (NSUB .EQ. 4 .AND. NBOX .EQ. 3) THEN
             CALL MONI_POP (3, PTXT, NPBOX)
C           *AND CONFIRMS THE DELETION
             IF (NPBOX .EQ. 1) THEN
                PRINT*,'DELETION OPTION SELECTED'
             ENDIF
          ENDIF
          GO TO 100
          STOP
          END
```

8. Summary. The Macro-FLIK routines described in the previous sections are intended to provide the application software programmer with an easy to use, minimal set of routines which will enable him to incorporate VAX/Ramtek graphics into any application package. Currently, the major graphics functions provided include menuing (graph tablet input), data point display and analysis, and data point manipulation and preservation. Other non-application software-specific graphics capabilities could be added to the current package upon request.

a. The Macro-FLIK software can be found in the disk/directory defined by logical name ØMACRO. The source code is separated into two text libraries. Library MACRO.TLB contains that portion of the package designed to be callable by the applications programmer, while library UTLY.TLB contains utility routines called by the routines in the former. The applications programmer calls utility routines at his own risk with no guarantee of the results. The object modules for both text libraries are kept in A.OLB.

b. Most of the sample applications programs found in this document (and several others) are available in ØMACRO:APPL.TLB for additional assistance.

c. At this writing, two additional enhancements to the Macro-FLIK package have been suggested and are under consideration. The first of these would involve the incorporation of "alert" menus. Alert menus are very similar to pop-up menus in that they appear only for a single selection; but, they differ in that they have space available for user provided text instruction and a number of "buttons" (boxes) for the user to select from. The second enhancement involves a simplistic method for displaying text of a user-specified color and size and to a user specified location on the monitor screen.

d. Any questions regarding the use of the Macro-FLIK package, or suggestions as to how to improve its utility, should be directed to Mr Pete Kaeding, TRAC-FLVN, AV 552-3193/4261.

DISTRIBUTION LIST

	No. Copies
Defense Technical Information Center ATTN: DTIC, TCA Cameron Station Alexandria, VA 22314	2
US Army Library Army Study Documentation and Information Retrieval System (ASDIRS) ANRAL-RS ATTN: ASDIRS Room 1a518, The Pentagon Washington, D.C. 20310	1
US Army TRADOC Analysis Command-WSMR ATTN: ATRC-WSL (Technical Library) White Sands Missile Range, NM 88002-5502	1
US Army TRADOC Analysis Command-FLVN ATTN: ATRC-FOA (Technical Info Center) Fort Leavenworth, KS 66027-5200	1
US Army Combined Arms Research Library (CARL) ATTN: ATZL-SWS-L Fort Leavenworth, KS 66027-5000	1